

Resumo de estudos

# Aulas de Back-End

João V. Montanari



# Desenvolvimento Web

# Projeto final

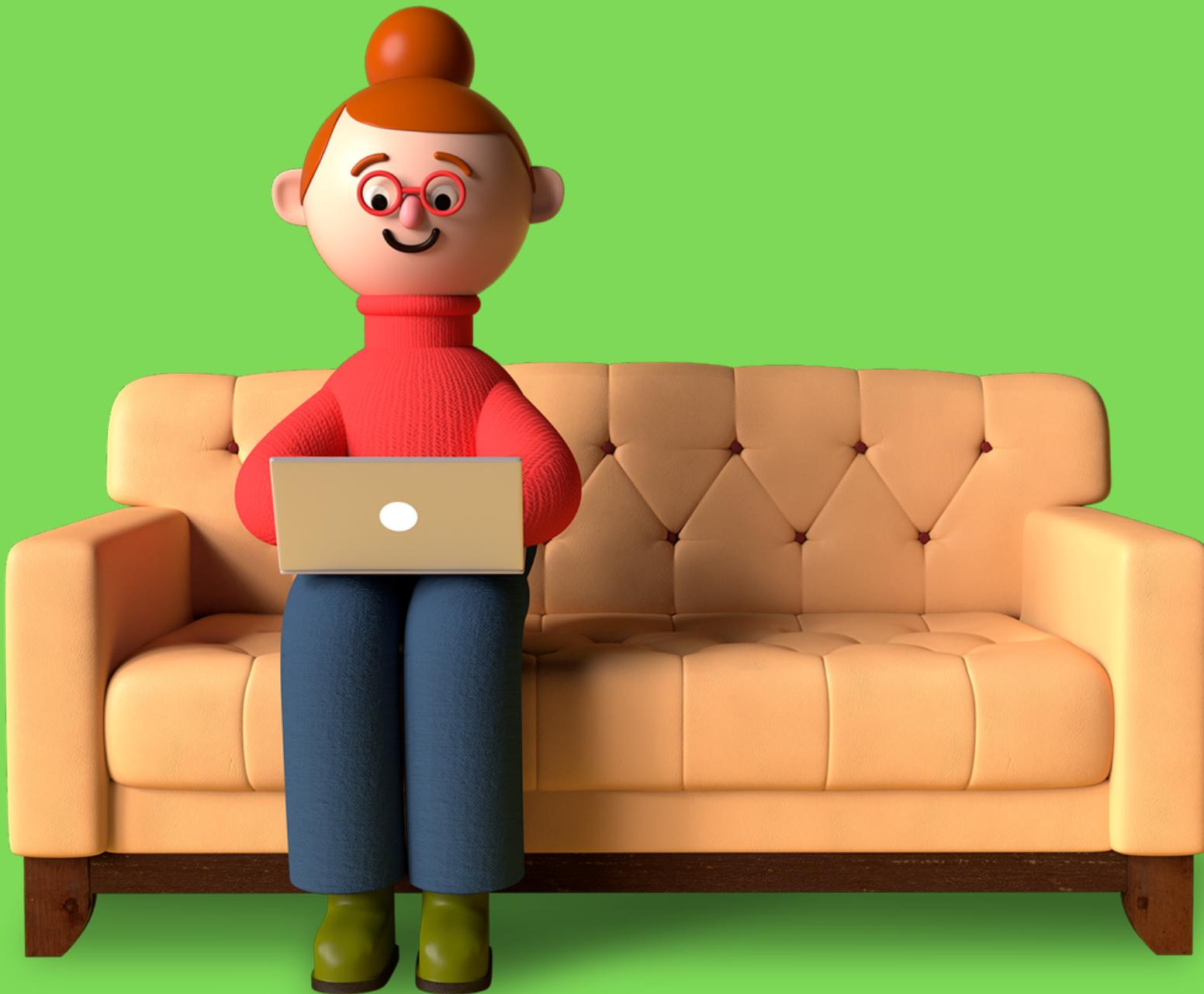
Desenvolvimento de um Banco:

- Aplicação web
- Aplicação mobile
- Banco de dados
- API



# Desenvolvimento web

## MÓDULO-01



Introdução ao desenvolvimento  
Back-End!  
Ferramentas e definições.



Back-end

# Ferramentas e definições:

Para o desenvolvimento de uma boa aplicação é preciso que a equipe de desenvolvedores tenham conhecimentos de ferramentas e conceitos bastante utilizados no mundo Back-End, tendo ciência disso vamos começar!



Back-end

# Ferramentas de compartilhamento:

Raramente um desenvolvedor vai estar trabalhando sozinho dentro de um projeto, logo é muito importante que o mesmo tenha conhecimento de ferramentas que ajudam no compartilhamento de códigos e ideias, ferramentas que permitem que diferentes membros de uma mesma equipe compartilhem do mesmo ambiente de desenvolvimento.



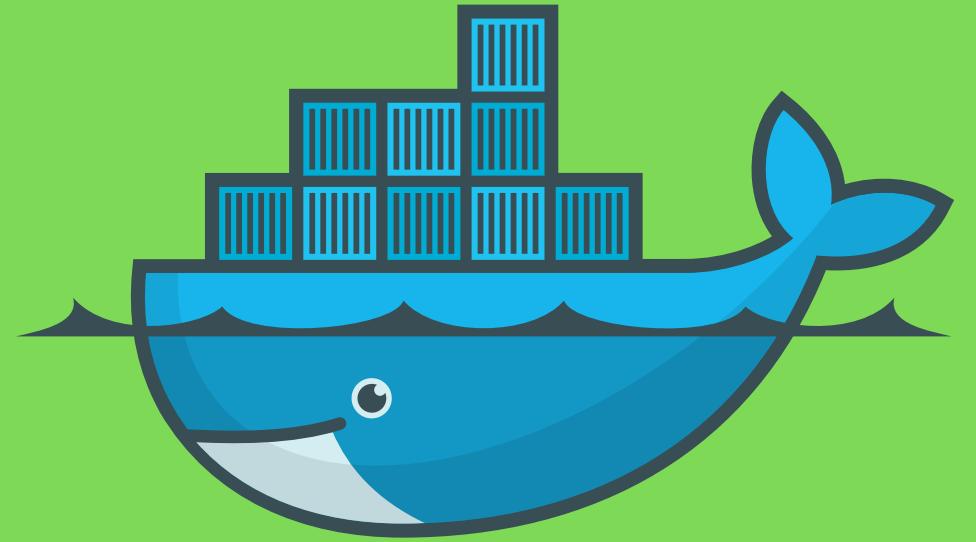
# Ferramentas de compartilhamento:

Assim para podermos tornar o projeto mais dinâmico e interativo entre os diferentes programadores utilizamos de ferramentas como:

Docker:

- O Docker atua como uma "máquina virtual" se tratando de uma instância dentro de sua máquina. Serve para que diferentes pessoas possam trabalhar no mesmo ambiente.

- Indo para uma definição mais precisa dizemos que o software utiliza de serviços de virtualização para poder entregar softwares em pacotes chamados contêineres. E esses mesmos contêineres são isolados uns dos outros e agrupam seus próprios softwares, bibliotecas e arquivos de configuração. Logo não é necessário que a máquina que vai ser utilizada para o desenvolvimento tenha tudo preparado pois o Docker vai conter com um contêiner com as configurações para rodar a aplicação.



# Ferramentas de compartilhamento:

Continuando, podemos citar outra grande plataforma para compartilhamento...

Git e GitHub:

- O Git é um sistema de controle de versão de arquivos. É um software livre utilizado no desenvolvimento de software onde diversas pessoas contribuem simultaneamente, podendo criar e editar arquivos. Quando alguém disponibiliza sua parte do projeto no Git, ele gerencia as alterações feitas e guarda um histórico.



- O GitHub é uma plataforma onde você pode armazenar seus projetos. É como se fosse uma rede social, só que de códigos, onde seus desenvolvedores podem disponibilizá-los para outras pessoas verem. Quando seu projeto está no GitHub, você pode facilmente baixar uma cópia em outro computador. É uma plataforma gratuita e armazena milhões de projetos, tanto open source, pessoais e até mesmo comerciais.

Back-end

# Ferramentas de compartilhamento:

Para conhecer mais dessas ferramenta acesse os links disponibilizados:

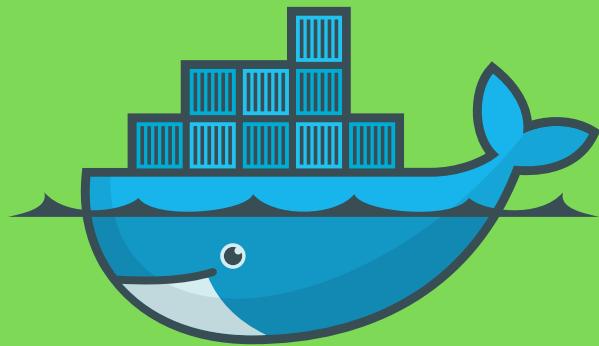
Site oficial Docker: <https://www.docker.com>

Site oficial GitHub: <https://github.com>

Curso completo de Git e GitHub: <https://youtu.be/xEKo29OWILE>

Instalar Git: <https://git-scm.com/downloads>

Instalar Docker: <https://docs.docker.com/desktop/install/windows-install/>



Back-end

# Ferramentas e definições:

Agora que já falamos de algumas ferramentas utilizadas no mundo da programação, vamos partir para alguns conceitos muito importantes para entender melhor esse universo tão vasto de desenvolvimento.



DevOps

# Definições:

O DevOps se trata de uma filosofia de trabalho dentro do mundo de desenvolvimento de programas, na onde todos que estão envolvidos no projeto tem que estar a par de todos os acontecimentos dele, assim a equipe não corre riscos de entregar um produto final que não atenda as expectativas do cliente!

Assim a filosofia garante que a emprese trabalhe de uma maneira mais rápida e eficiente, o que permite que a mesma se adeque melhor a um mercado mais dinâmico e entregue rapidamente o produto.



DevOps

# Definições:

Os benefícios da filosofia DevOps:



**Velocidade**



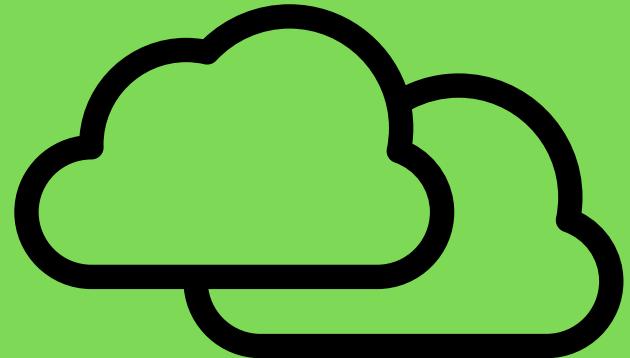
Opere em alta velocidade para que você possa trazer inovações para os seus clientes mais rapidamente, adapta se melhor a mercados dinâmicos e tornar-se mais eficiente na geração de resultados comerciais. O modelo de DevOps permite que as suas equipes de desenvolvedores e operações atinjam esses resultados. Por exemplo, os microserviços e a entrega contínua permitem que as equipes assumam a responsabilidade sobre os serviços e, então, lancem atualizações para eles mais rapidamente.

DevOps

# Definições:

Os benefícios da filosofia DevOps:

Entrega rápida



Aumenta a frequência e o ritmo de lançamentos para poder inovar e melhorar seu produto mais rapidamente. Quanto mais rápido você puder lançar novos recursos e corrigir erros, maior será a sua agilidade para responder às necessidades dos clientes e criar vantagem competitiva. A integração e a entrega contínuas são práticas que automatizam o processo de lançamento de software, da fase de criação à fase de implantação.



DevOps

# Definições:

Os benefícios da filosofia DevOps:



## Confiabilidade



Garanta a qualidade das atualizações de aplicativos e alterações de infraestrutura para que você possa entregar com confiança em um ritmo mais rápido, sem deixar de manter uma experiência positiva para os usuários finais. Use práticas como a integração e a entrega contínuas para testar se cada uma das alterações funciona e é segura. As práticas de monitoramento e registro em log ajudam você a permanecer informado sobre a performance em tempo real.

DevOps

# Definições:

Os benefícios da filosofia DevOps:



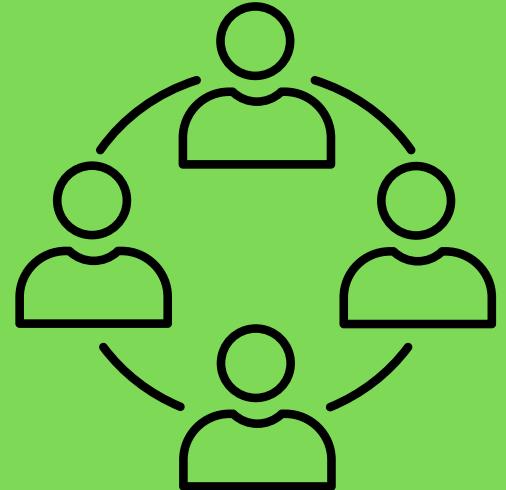
Opere e gerencie seus processos de infraestrutura e desenvolvimento em escala. A automação e a constância ajudam você a gerenciar sistemas complexos ou dinâmicos com eficiência e risco reduzido. Por exemplo, a infraestrutura como código ajuda você a gerenciar seus ambientes de implantação, teste e produção de modo repetido e mais eficiente.

DevOps

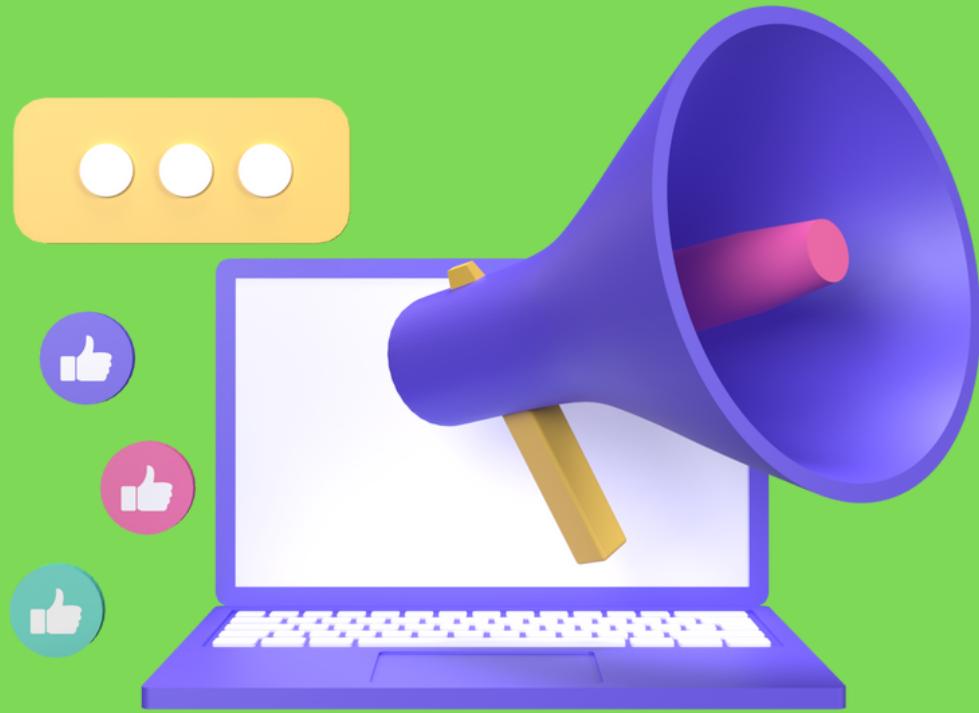
# Definições:

Os benefícios da filosofia DevOps:

## Colaboração



Crie equipes mais eficientes em um modelo cultural de DevOps, que enfatiza valores como propriedade e responsabilidade. As equipes de desenvolvedores e operações colaboram de perto, compartilham muitas responsabilidades e combinam seus fluxos de trabalho. Isso reduz ineficiências e economiza tempo (por exemplo, períodos de transferência reduzidos entre desenvolvedores e operações, desenvolvimento de código que considera o ambiente em que é executado).



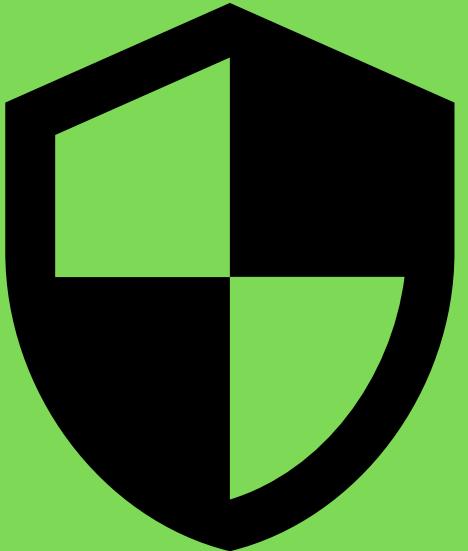
DevOps

# Definições:

Os benefícios da filosofia DevOps:



## Segurança

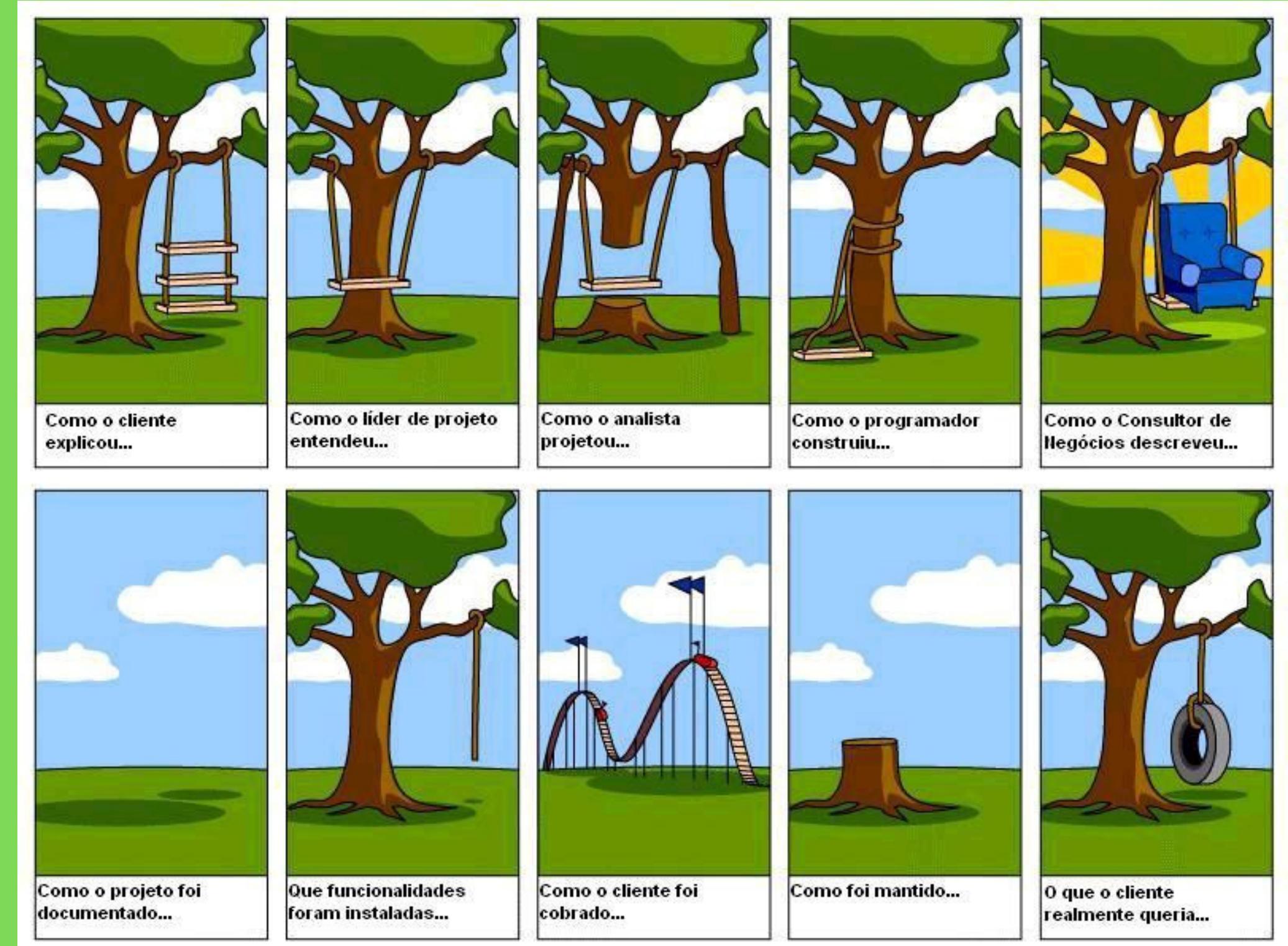


Opere rapidamente enquanto mantém o controle e preserva a conformidade. Você pode adotar o modelo de DevOps sem sacrificar a segurança usando políticas de conformidade automáticas, controles minuciosos e técnicas de gerenciamento de configuração. Por exemplo, usando a infraestrutura e a política como código, você pode definir e acompanhar a conformidade em escala.

# DevOps Definições:

Exemplificação em quadrinhos:

A filosofia DevOps existe para poder evitar situações como a exemplificada no quadrinho!



Padrão CI e CD

# Definições:

Continuando com algumas definições vamos partir agora para os padrões CI/CD, que se tratam respectivamente dos termos Continuous Integration e Continuous Delivery, juntos se tornam um método que visa a entrega de aplicações com frequência para os clientes.

O método tem como principais conceitos a integração, entrega e implementação contínua.

Especificamente, o CI/CD aplica monitoramento e automação contínuos em todo o ciclo de vida das aplicações, incluindo as etapas de teste e integração, além da entrega e implantação. E são compatíveis com o trabalho conjunto das equipes de operações e desenvolvimento com métodos ágeis.



Padrão CI e CD

# Definições:

Diferença entre CI e CD:

## CI

Sempre se refere à integração contínua, que é um processo de automação para desenvolvedores. Uma CI bem-sucedida é quando novas mudanças no código de uma aplicação são desenvolvidas, testadas e consolidadas regularmente em um repositório compartilhado. É a solução ideal para evitar conflitos entre ramificações quando muitas aplicações são desenvolvidas ao mesmo tempo.

## CD

Se refere à entrega contínua e/ou à implantação contínua, conceitos relacionados e usados alternadamente às vezes. Em ambos os casos, se trata da automação de fases avançadas do pipeline, mas são usados às vezes separadamente para ilustrar o nível de automação presente.



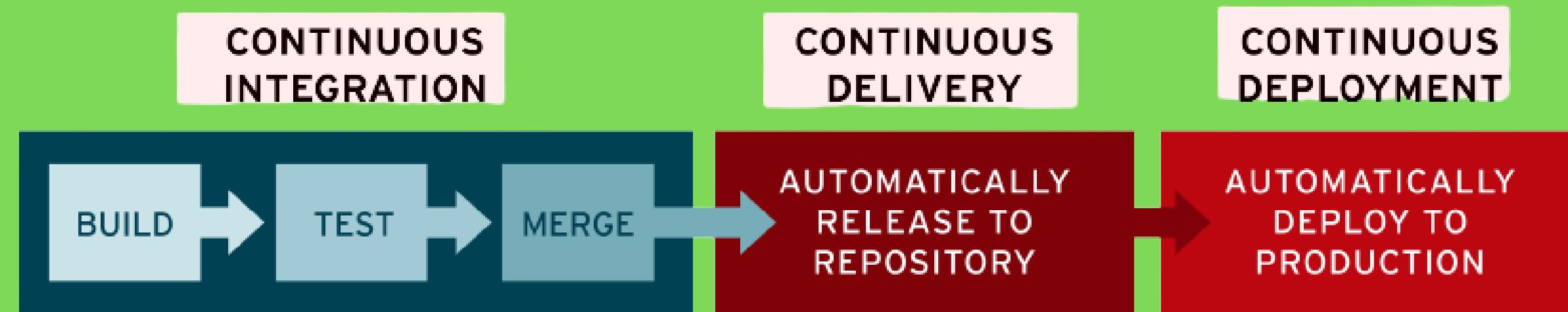
# Padrão CI e CD

## Definições:



Diferença entre CI e CD:

Geralmente, a entrega contínua representa as mudanças feitas pelo desenvolvedor em uma aplicação, que são automaticamente testadas contra bugs e carregadas em um repositório, como o GitHub, ou em um registro de container. Nesse repositório, a equipe de operações pode implantar essas mudanças em um ambiente de produção ativo. Isso resolve o problema de baixa visibilidade e comunicação entre as equipes de negócios e desenvolvimento. Para isso, a finalidade da entrega contínua é garantir o mínimo de esforço na implantação de novos códigos.



Padrão CI e CD

# Definições:

Integração continua:



No desenvolvimento moderno de aplicações, o objetivo é que muitos desenvolvedores trabalhem ao mesmo tempo em diferentes recursos na mesma aplicação.

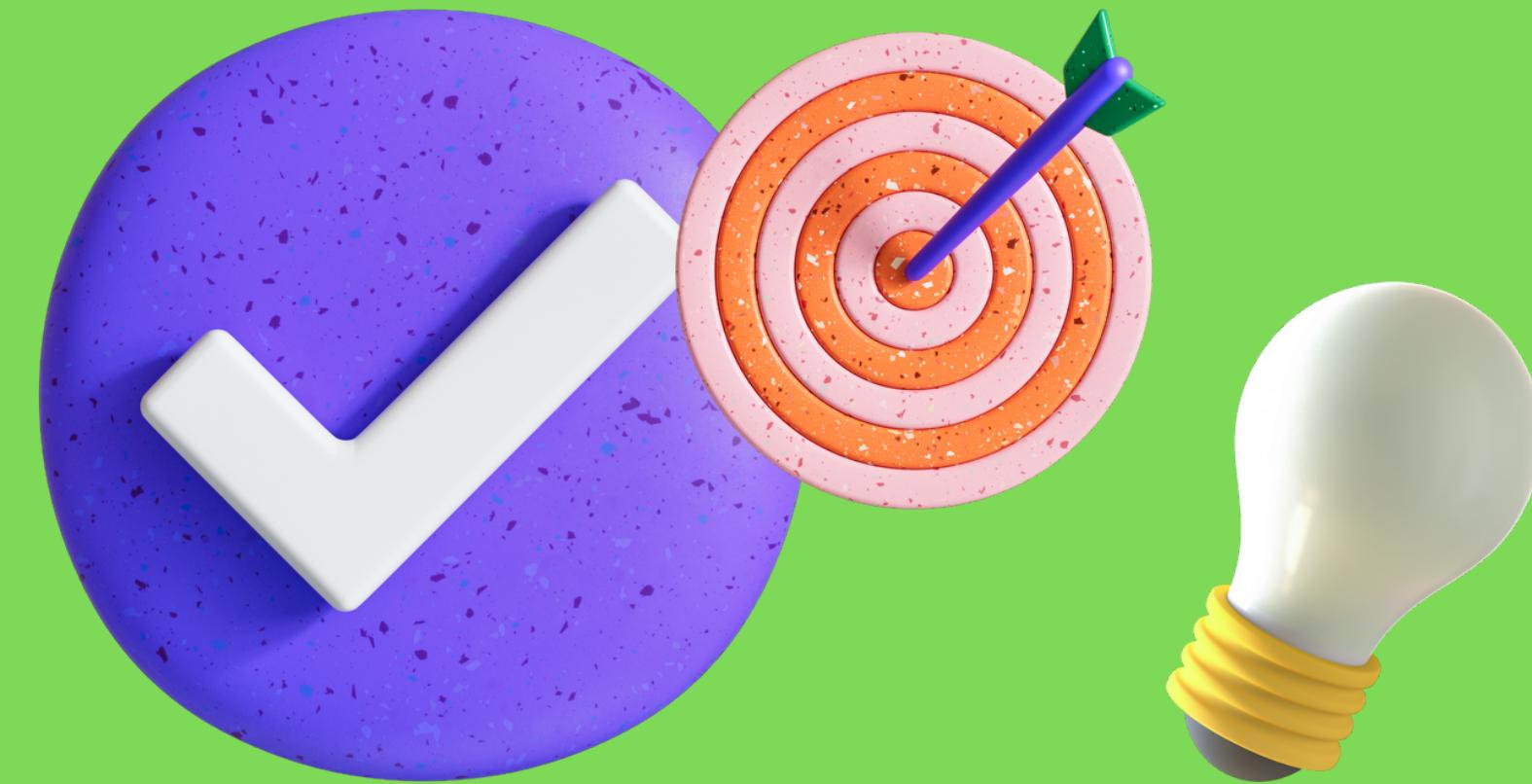
Com a integração contínua (CI), os desenvolvedores consolidam as mudanças no código de volta a uma ramificação compartilhada ou "tronco" com mais frequência (às vezes, até diariamente). As mudanças são consolidadas e depois validadas através da criação automática da aplicação. Vários testes automatizados, geralmente de unidade e integração, são feitos para garantir que as mudanças não corrompam a aplicação. Basicamente, tudo é testado, incluindo classes, funções e diferentes módulos que formam toda a aplicação. Em caso de conflito entre os códigos novos e existentes, a CI facilita a correção desses bugs com rapidez e frequência.

Padrão CI e CD

# Definições:

Entrega continua:

Depois de realizar a automação de compilações e da unidade e os testes de integração na CI, a entrega contínua automatiza o lançamento desse código validado em um repositório. Portanto, para ter um processo eficaz de entrega contínua, é importante que a CI já esteja integrada ao pipeline de desenvolvimento. O objetivo da entrega contínua é garantir uma base de códigos que esteja sempre pronta para implantação em um ambiente de produção.



# Padrão CI e CD

## Definições:

Implantação continua:



A etapa final de um pipeline de CI/CD sólido é a implantação contínua. Ela é um complemento da entrega contínua, que automatiza o lançamento de compilações prontas para produção em um repositório de códigos. A implantação contínua automatiza o lançamento de uma aplicação para a produção.

Na prática, a implantação contínua significa que a mudança do desenvolvedor em uma aplicação será habilitada depois de alguns minutos após a gravação. Isso facilita muito mais o recebimento do feedback dos usuários e a incorporação dele. Juntas, todas essas práticas de CI/CD relacionadas diminuem o risco da implantação de aplicações, facilitando o lançamento das mudanças em pequenas partes, e não de uma só vez.

Scrum

# Definições:

Agora vamos falar de uma estrutura de trabalho em equipe chamada Scrum que é bastante usada pelas equipes de desenvolvimento de software, mesmo assim, os princípios e as lições dessa estrutura podem ser aplicados a todos os tipos de trabalhos em equipe.

Muitas vezes considerado uma estrutura de gestão de projetos de agilidade, o Scrum descreve um conjunto de reuniões, ferramentas e cargos que atuam juntos para ajudar as equipes a organizarem e gerenciarem o trabalho.



Scrum

# Definições:

Estrutura:

Em geral, as pessoas pensam que o Scrum e a agilidade são a mesma coisa porque o Scrum é centrado na melhoria contínua, que é o princípio fundamental da agilidade. No entanto, o Scrum é uma estrutura para concluir tarefas, enquanto a agilidade é uma forma de pensar.

A estrutura do Scrum é heurística; ela é baseada no aprendizado contínuo e na adaptação aos fatores variáveis. O Scrum reconhece que a equipe não sabe tudo no início de um projeto e que evoluirá de acordo com a experiência. Ele é estruturado para ajudar as equipes a se adaptarem naturalmente às mudanças e aos requisitos do usuário, com repriorização integrada no processo e ciclos curtos de liberação para que sua equipe aprenda e melhore constantemente.



Scrum

# Definições:

Artefatos Scrum:



Um artefato é algo que produzimos, como uma ferramenta para resolver problemas. No Scrum, os três artefatos são um backlog do produto, um backlog do sprint e um incremento com aquilo que você define como "concluído". Eles são as três constantes em uma equipe do Scrum que continuam sendo revisitadas



Scrum

# Definições:

Artefatos Scrum:

**Backlog do produto**



É a principal lista do trabalho que precisa ser feita e é mantida pelo proprietário do produto ou gerente de produtos. É uma lista dinâmica de recursos, requisitos, aprimoramentos e correções que atua como a entrada para o backlog do sprint. Basicamente, ela é a "Lista de afazeres" da equipe. O backlog do produto é sempre revisto, repriorizado e mantido pelo proprietário do produto porque, conforme aprimoramos o conhecimento ou o mercado muda, os itens podem não ser mais relevantes ou os problemas podem ser resolvidos de outras formas.



Scrum

# Definições:

Artefatos Scrum:

**Backlog do sprint**



É a lista de itens, histórias de usuários ou correções de bugs selecionada pela equipes de desenvolvimento para a implementação no ciclo atual de sprint. Antes de cada sprint, durante a reunião de planejamento de sprint, a equipe escolhe quais itens funcionarão para o sprint a partir do backlog do produto. Um backlog do sprint pode ser flexível e se desenvolver durante um sprint. No entanto, a meta fundamental do sprint, ou seja, o que a equipe deseja alcançar com o sprint atual, não pode ser comprometida.



Scrum

# Definições:

Artefatos Scrum:

**Incremento**



É o produto final utilizável, proveniente de um sprint. Talvez você não escute a palavra "incremento" por aí, visto que ela costuma ser citada como a definição de "Concluído" dada pela equipe, como um marco, a meta de sprint ou, até mesmo, uma versão completa ou um epic lançado. Depende apenas de como as equipes definem "Concluído" e como você define suas metas de sprint. Por exemplo, algumas equipes optam por lançar algo para seus clientes no final de cada sprint. Dessa forma, para elas, a definição de "Concluído" pode ser "lançado". No entanto, essa definição pode não ser plausível para outros tipos de equipes.



Scrum

# Definições:

Atividades e planejamentos:

Dentro do Scrum existem certas atividades que são realizadas para poder manter a integridade do projeto e definir as atividades que vão se proceder no desenvolvimento!



Scrum

# Definições:

Artefatos Scrum:



## Organizar o Backlog



Algumas vezes conhecido como "preparação do backlog", esse evento é responsabilidade do proprietário do produto. As principais tarefas do proprietário é orientar o produto em direção à visão do produto e acompanhar constantemente o mercado e o cliente. Dessa forma, ele mantém a lista usando o feedback dos usuários e da equipe de desenvolvimento para ajudar a priorizar e manter a lista clara e pronta para ser trabalhada a qualquer momento.

Scrum

# Definições:

Artefatos Scrum:



## Planejamento de sprints



O trabalho que será realizado (escopo) ao longo do sprint atual é planejado durante essa reunião por toda a equipe de desenvolvimento. A reunião é conduzida pelo mestre do Scrum e é nela que a equipe decide a meta de sprint. Histórias de uso específicas são, então, acrescentadas ao sprint a partir do backlog do produto.

No final da reunião de planejamento, cada membro do Scrum precisa esclarecer o que pode ser apresentado no sprint e como o incremento pode ser entregue.

Scrum

# Definições:

Artefatos Scrum:

**Sprint**

Um sprint é o período real em que a equipe do Scrum trabalha em conjunto para concluir um incremento. Dave West, da Scrum.org, adverte que, quanto mais complexo e incerto for o trabalho, menor deve ser o sprint. Mas esse período fica realmente a critério da sua equipe, e você não deve ter medo de mudá-lo se não estiver funcionando. Durante essa fase, o escopo pode ser renegociado entre o proprietário do produto e a equipe de desenvolvimento, se necessário. Isso constitui a essência da natureza empírica do Scrum.

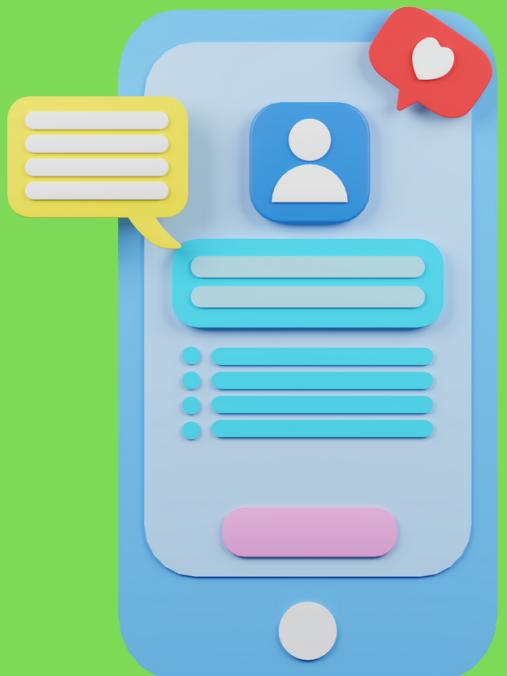
Scrum

# Definições:

Artefatos Scrum:



**Scrum diário ou  
reunião diária**

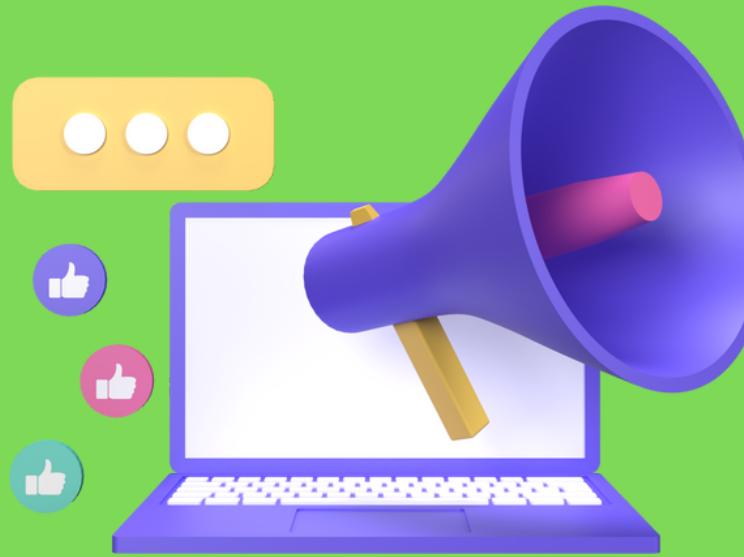


A meta do Scrum diário é fazer com que todos os integrantes da equipe estejam atualizados com as mesmas informações e alinhados com a meta do sprint para chegarem a um planejamento para as próximas 24 horas.

Scrum

# Definições:

Artefatos Scrum:



## Análise de sprint



No final do sprint, a equipe se reúne para uma sessão informal a fim de ver uma demonstração do incremento ou inspecioná-lo. A equipe de desenvolvimento mostra os itens de backlog que estão "concluídos" para às partes interessadas e aos colegas de equipe para que eles possam dar o feedback. O proprietário do produto pode decidir se vai lançar ou não o incremento.

Scrum

# Definições:

Artefatos Scrum:



## Retrospectiva de sprint



A retrospectiva é o momento em que a equipe se reúne para documentar e discutir o que funcionou e o que não funcionou em um sprint, em um projeto, nas pessoas ou nos relacionamentos, nas ferramentas ou, até mesmo, em determinadas cerimônias. A ideia é criar um local em que a equipe possa focar o que foi bem e o que precisa melhorar para a próxima vez.

Padrão MVC

# Definições:

O que é:

Agora vamos aprender um pouco do padrão MVC. O MVC é uma sigla do termo em inglês Model (modelo) View (visão) e Controller (Controle) que facilita a troca de informações entre a interface do usuário aos dados no banco, fazendo com que as respostas sejam mais rápidas e dinâmicas.



# Padrão MVC

# Definições:

O que é:



**Model**



Essa classe também é conhecida como Business Object Model (objeto modelo de negócio). Sua responsabilidade é gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas.

Ele é o detentor dos dados que recebe as informações do Controller, válida se ela está correta ou não e envia a resposta mais adequada.

# Padrão MVC

# Definições:

O que é:



## Controller



A camada de controle é responsável por intermediar as requisições enviadas pelo View com as respostas fornecidas pelo Model, processando os dados que o usuário informou e repassando para outras camadas.  
Numa analogia bem simplista, o controller operaria como o "maestro de uma orquestra" que permite a comunicação entre o detentor dos dados e a pessoa com vários questionamentos no MVC.

# Padrão MVC

# Definições:

O que é:

**View**



Essa camada é responsável por apresentar as informações de forma visual ao usuário. Em seu desenvolvimento devem ser aplicados apenas recursos ligados a aparência como mensagens, botões ou telas.

Seguindo nosso processo de comparação o View está na linha de frente da comunicação com usuário e é responsável transmitir questionamentos ao controller e entregar as respostas obtidas ao usuário. É a parte da interface que se comunica, disponibilizando e capturando todas as informações do usuário.



Padrão MVC

# Definições:

Como os componentes interagem:

Tudo começa com a interação do usuário na camada View. A partir daí o controlador pega essa informações e envia para o Model que fica responsável por avaliar aqueles dados e transmitir uma resposta.

O controlador recebe essas respostas e envia uma notificação de validação daquela informação para a camada visão, fazendo com a mesma apresente o resultado de maneira gráfica e visual.

Todo esse processo leva em consideração as regras de negócio aplicadas na construção de todo projeto



# Padrão MVC

## Definições:

### Por que usar?

Muitos bootcamps de programação ensinam esse padrão de arquitetura de software por alguns benefícios que justificam o MVC como uma das mais escolhidas no processo de desenvolvimento.

Esses benefícios são:

- **Segurança:** O controller funciona como uma espécie de filtro capaz de impedir que qualquer dado incorreto chegue até a camada modelo.

- **Organização:** Esse método de programação permite que um novo desenvolvedor tenha muito mais facilidade em entender o que foi construído, assim como os erros se tornam mais fácil de serem encontrados e corrigidos.
- **Eficiência:** Como a arquitetura de software é dividida em 3 componentes , sua aplicação fica muito mais leve, permitindo que vários desenvolvedores trabalhem no projeto de forma independente.
- **Tempo:** Com a dinâmica facilitada pela colaboração entre os profissionais de desenvolvimento, o projeto pode ser concluído com muito mais rapidez, tornando o projeto escalável.

Padrão MTV

# Definições:

O que é:

A arquitetura MTV é uma derivação do MVC padrão de projeto. O que se altera é a nomenclatura de arquivos e quais as camadas interconectadas. Este modelo é utilizado dentro do framework Django.



Padrão MTV

# Definições:

O que é:



Model



A definição de Model é a mesma para a arquitetura MVC e MTV, que é o arquivo que contém a estrutura lógica do projeto e funciona como um intermediário para manipular dados entre o banco de dados e a View. Dentro do arquivo Model é determinado quais tipos de dados, e como será armazenado dentro do seu banco e, como será exibido quando for requisitado pela View.

Padrão MTV

# Definições:

O que é:



**Controller**



O arquivo View é muito confundido com o Controller da arquitetura MVC, porém não tem relação. O papel desta camada é formatar os dados que são vindos do banco através da Model para visualização.

# Padrão MTV

# Definições:

O que é:

**View**



E por último o Template, que cuida da parte desta visualização para o usuário final. Ele é como o front-end de sua aplicação. Nesta arquitetura, esta camada fica armazenado os arquivos html, css, javascript extendidos e por conta disso auxilia numa velocidade maior de desenvolvimento e conforme o retorno da aplicação, ele renderiza seus arquivos HTML de sua aplicação no navegador. Como referência, esta camada seria como a camada View em uma estrutura MVC, e no caso do Django framework, tem um plus que é possível trabalhar com extensão de arquivos, ou seja, para o dev front-end isso auxilia muito em seu trabalho.



Back-end

# Definições:

Para saber mais acesse os links abaixo:

Padrão MTV: <https://diandrasilva.medium.com/como-funciona-a-arquitetura-mtv-django-86af916f1f63>

Padrão MVC: <https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc>

SCRUM: <https://www.atlassian.com/br/agile/scrum>

Padrão CI/CD: <https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd#entrega-contínua>



# Me siga nas redes sociais:

Espero que tenha gostado!

Aproveite e me siga nas redes sociais :)



@joaomontanari26



joao-montanari



João V. Montanari